A Serial Complexity Measure of Neural Networks

Moshe Sipper
Department of Computer Science
School of Mathematical Sciences
Sackler Faculty of Exact Sciences
Tel Aviv University
69978, Tel Aviv, Israel
e-mail: moshes@math.tau.ac.il

***Abstract*—**

The most common methodology of neural network analysis is that of simulation since as of yet there is no common formal framework. Towards this end we adopt one measure of serial algorithms, namely that of serial computational complexity and apply it to the analysis of neural networks. We analyze various networks and derive their complexity, thus providing insight as to their computational requirements.

I. Introduction

The most common methodology used for the purpose of demonstrating a neural network's effectiveness is that of simulation. This entails verifying the proposed network's performance in an empirical setting rather than from a theoretical standpoint. More rigorous analysis' have been carried out, yielding improved results as to various aspects of neural networks such as convergence rates and storage capacities. As of yet there is no common framework for analyzing the effectiveness of neural networks.

Towards this end we adopt one measure of serial algorithms, namely that of serial computational complexity and apply it to the analysis of neural networks. While such an analysis ignores the parallelism issues inherent in neural networks, it nevertheless provides us with a picture of the computational complexity of a given model. Thus such a measure may serve as a guideline for implementation and comparison.

We use the ubiquitous RAM (Random Access Machine) model [1] which may be described in simple terms as a Turing machine with a RAM (Random Access Memory). The instructions in this model are executed sequentially, unless control flow is altered by the execution of a branch. A common instruction set is used [1] where addition, subtraction and multiplication are included (among others) as elementary operations. Note however that the weighted sum of the artificial neuron requires a succession of such elementary steps.

II. Serial Complexity of Neural Networks

In this section we analyze various networks and derive their serial computational complexity.

A. The Hamming network

The Hamming network calculates the Hamming distance between the input pattern and each memory pattern, and selects the memory with the smallest distance, which is declared 'the winner'. This network is the most straightforward associative memory. Originally presented in [16, 17, 18], it has received renewed attention

0-7803-0999-5/93/$03.00 ©1993 IEEE

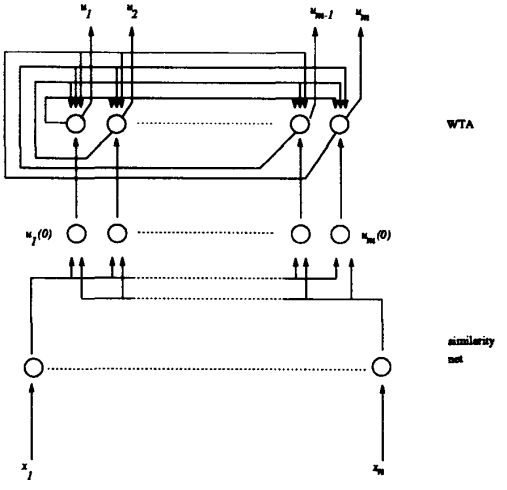

Figure 1: **The Hamming network**

in recent years [12, 2, 10]. The Hamming network operates on binary vectors of ± 1 and is depicted in Figure 1.

It is composed of two subnets. The lower subnet, denoted the *similarity* subnet calculates the Hamming distance between the input vector and each memory pattern. It consists of two layers: An n-neuron input layer representing n-bit input patterns, and an m-neuron memory layer where each neuron represents one memory. Memory storage is achieved via the connection weights entering the neuron. The upper subnet, denoted as the *winner-take-all (WTA)*, computes the memory which is at minimum Hamming distance from the input. It consists of a fully connected m-neuron topology. The similarity subnet is feedforward whereas the WTA subnet is iterative.

The initialization phase of the Hamming network consists of assigning weights in the following manner [10]:

In the lower subnet:

$w_{ij} = x_i^j/2 \qquad \Theta_j = n/2$

$1 \leq i \leq n, 1 \leq j \leq m$

w_{ij} is the connection weight from node i to node j in the lower subnet.

Θ_j is the threshold in that node.

x_i^j is element i of exemplar j.

In the upper subnet:

$$t_{kl} = \begin{cases} 1 & \text{k = l} \\ -\epsilon & k \neq l \text{ , } \epsilon < 1/\text{m} \end{cases}$$

t_{kl} is the connection weight from node k to node l in the upper subnet.

All thresholds are zero in this subnet.

The total complexity of this phase is $O(mn + m^2)$.

The Run phase consists of iterating until convergence:

$$u_j(t+1) = f_t(u_j(t) - \epsilon \sum_{k \neq j} u_k(t))$$

$1 \leq j, k \leq m$

$u_j(t)$ is the output of node j in the upper network, at time t.

f_t is a threshold logic function.

The process is repeated until convergence. At this time only one node remains positive.

The total worst case parallel running time has been shown to be $O(m \ln(mn))$ [5]. The serial complexity is therefore $O(m^3 \ln(mn))$ and thus the total serial run time complexity is $O(mn + m^3 \ln(mn))$.

The total serial complexity (initialization phase + run phase) of the Hamming network is therefore $O(mn + m^3 \ln(mn))$.

B. The Hopfield network

The Hopfield network ([7]) is among the most commonly researched neural networks and has been analyzed in depth. We examined an n-neuron, fully connected network. The initialization phase consists of assigning synaptic weights [10]:

$$t_{ij} = \begin{cases} \sum_{s=1}^{S} x_i^s x_j^s & i \neq j \\ 0 & \text{i=j} \end{cases}$$

S is the sample set size.

$1 \leq i, j \leq n$

t_{ij} is the synaptic weight from node i to node j.

x_i^s is element i of class s exemplar and is binary valued ("1" or "-1").

The complexity of this phase is $O(Sn^2)$.

The Run phase consists of iterating until convergence:

$$u_j(t+1) = f_h(\sum_{i=1}^{n} t_{ij} u_i(t))$$

$1 \leq j \leq n$

$u_i(t)$ is the output of node i at time t.

f_h is the hard limiter function.

The process is repeated until convergence. Convergence is defined as the time at which node outputs remain unchanged.

The parallel running time has been shown to be $O(\log(\log n))$ [9]. Thus the total serial complexity of the run phase is: $O(n^2 \log(\log n))$.

The total serial complexity of the Hopfield network is therefore $O(Sn^2 + n^2 \log(\log n))$.

C. A multi-layer, back-propagation trained perceptron

One of the most common models of neural networks is that of a multi-layer back-propagation trained perceptron ([15]). This model has received wide attention ([8, 11, 10]) and has been studied extensively. We consider a three layer perceptron with n neurons in the input layer, l neurons in the hidden layer and m neurons in the output layer.

The complexity of the backpropagation network is due entirely to the learn phase, which is iterative. A forward pass is of complexity $O(l(n+m))$ while a backward pass which involves the error computations is $O(lnm)$ and thus the total complexity is $O(lnm)$ for a *single* forward and backward pass. As of yet there is no formal result as to the convergence rate.

D. A second order network

High order networks, which replace the linear neuron with a polynomial of the form $\{w_i + \sum_j w_{ij}x_j + \sum_{jk} w_{ijk}x_j x_k + \ldots\}$ have been an object of research in recent years. It was demonstrated that such networks achieve improved learning rates ([6, 13]) and increased storage capacity ([14, 3]). In many cases it is easier to train a high order network than a multi-layer network since training the hidden layers is more difficult ([13]).

We examined a second order, fully connected, two layer network composed of an n-neuron input layer and an m-neuron output layer. The initialization phase consists of a 'one-shot' Hebbian rule [6] and its complexity is $O(Smn^2)$ where S is the size of the sample set. The run phase consists of a single feedforward sweep and is $O(mn^2)$. Thus the total complexity of the network is $O(Smn^2)$.

III. Conclusions

Table 1 summarizes our findings of the previous section. We can compare the complexities of the backpropagation network, for a given learn time of, say TI_{bap}, with that of the second order network. Thus, $O(Smn^2) < O(TI_{bap}lnm)$ yields:

$$O(l) > O(\frac{Sn}{TI_{bap}})$$

Such a comparison reveals the point at which it is more efficient to use a two-layered second order network instead of a three-layered first order network. This point occurs when the number of hidden layers exceeds the number given by the above expression.

It is also possible to compare the Hopfield model and the Hamming model. In an optimal setting (i.e. where the network's memory capacity is not exceeded) $n = O(\log m)$ for the Hamming network [4] and $n = O(m)$, $S = O(m)$ for the Hopfield network [9]. Thus the Hamming serial complexity is $O(m^3 \log(m \log m))$ and the Hopfield serial complexity is $O(m^3)$, so that the Hopfield network's serial complexity is lower (i.e. better).

The main motivation of the analysis carried out in this paper has been to derive a formal

	Serial Complexity
The Hamming network	$O(mn + m^3 \ln(mn))$
The Hopfield network	$O(Sn^2 + n^2 \log(\log n))$
Back-propagation network (one pass)	$O(lnm)$
A second order network	$O(Smn^2)$

Table 1: Analysis results of the networks in the previous section

computational complexity measure of neural networks. Although in essence we measure the complexity of the serial implementation we feel that insight is gained as to the computational requirements of the various networks. It is hoped that other measures will be devised which will be used in the formal analysis of neural networks.

References

[1] G.S. Almasi and A. Gottlieb. *Highly Parallel Computing*. The Benjamin/Cummings Publishing Company Inc., 1989.

[2] E.E. Baum, J. Moody, and F. Wilczek. Internal representations for associative memory. *Biological Cybernetics*, 59:217–228, 1987.

[3] H.H. Chen, Y.C. Lee, G.Z. Sun, H.Y. Lee, T. Maxwell, and C.L. Giles. High order correlation model for associative memory. In J.S. Denker, editor, *AIP Conference Proceedings 151: Neural Networks for Computing*, pages 86–99, Snowbird, Utah, 1986.

[4] E. Domany and H. Orland. A maximum overlap neural network for pattern recognition. *Physics Letters A*, 125(1):32–34, 1987.

[5] P. Floréen. The convergence of Hamming memory networks. *IEEE Trans. Neural Networks*, 2(4):449–457, July 1991.

[6] C.L. Giles and T. Maxwell. Learning, invariance, and generalization in high order neural networks. *Applied Optics*, 26. No. 23:4972–4978, 1987.

[7] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA*, 79:2554–2558, April 1982.

[8] T. Khanna. *Foundations of Neural Networks*. Addison-Wesley Publishing Company, 1990.

[9] J. Komlós and R. Paturi. Effect of connectivity in associative memory models. In *Proc. of the 29th IEEE Annual Symp. on Foundations of Computer Science*, pages 138–147, 1988.

[10] R.P. Lippmann. An introduction to computing with neural nets. *IEEE ASSP magazine*, pages 4–22, 1987.

[11] R.P. Lippmann. Pattern classification using neural networks. *IEEE Communications Magazine*, pages 47–64, November 1989.

[12] R.P. Lippmann, B. Gold, and M.L. Malpass. A comparison of Hamming and Hopfield neural nets for pattern classification. Technical Report TR-769, MIT Lincoln Laboratory, 1987.

[13] T. Maxwell, C.L. Giles, Y.C. Lee, and H.H. Chen. Nonlinear dynamics of artificial neural systems. In J.S. Denker, editor, *AIP Conference Proceedings 151: Neural Networks for Computing*, pages 299–305, Snowbird, Utah, 1986.

[14] P. Peretto and J.J. Niez. Long term memory storage capacity of multiconnected neural networks. *Biol. Cybern.*, 54:53–63, 1986.

[15] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart, J.L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing, Volume 1: Foundations*, pages

318–362. The MIT Press, Cambridge, MA., 1986.

[16] K. Steinbuch. Dei lernmatrix. *Kybernetic*, 1:36–45, 1961.

[17] K. Steinbuch and U.A.W. Piske. Learning matrices and their applications. *IEEE Transactions on Electronic Computers*, pages 846–862, 1963.

[18] W.K. Taylor. Cortico-thalamic organization and memory. *Proc. of the Royal Society of London B*, 159:466–478, 1964.